You would like to set a password for an email account. However, there are two restrictions on the format of the password. It has to contain at least one uppercase character and it cannot contain any digits.

You are given a string S consisting of N alphanumerical characters. You would like to find the longest substring of S that is a valid password. A substring is defined as a contiguous segment of a string.

For example, given "a0Ba", the substrings that are valid passwords are "B" and "Ba". Note that "aBa" is not a substring and "a0B" is not a valid password.

Write a function:

```
object Solution { def solution(s: String): Int }
```

that, given a non-empty string S consisting of N characters, returns the length of its longest substring that is a valid password. If there is no such substring, your function should return −1.

For example, given "a0Ba", your function should return 2, as explained above. Given "a0bb", your function should return −1, since there is no substring that satisfies the restrictions on the format of a valid password.

Assume that:

- N is an integer within the range [1..200];
- string S consists only of alphanumerical characters (a−z and/or A−Z and/or 0−9).

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

Link to this task: codility.com/tasks/digitless_password/